Multi-Scale Location Analysis of Vulnerabilities and Their Link to Disturbances within Digital Ecosystems

Jennifer Jackson, Complexity Science, University of Warwick, February 2017

Keywords: Cyber security, computing, vulnerabilities, exploits, malware, security attacks

Summary

As computer networks evolve, so too does the techniques used by attackers to exploit new vulnerabilities. Natural ecosystems already have resistant and resilient properties that help protect them from unwanted disturbances despite the existence of different vulnerabilities. Computer networks and their environments can be considered as *digital* ecosystems with different vulnerabilities, and security attacks can be considered as unwanted disturbances. Analysis of vulnerabilities and attacks from this perspective may open up new ecosystem-based security strategies.

This study therefore considers computer networks from an ecosystem perspective with a focus on the location of vulnerabilities at different scales and their link to undesirable security disturbances. The location of vulnerabilities are considered from a physical, functional and time perspective; all aspects which are important within ecosystems and hence also important for defining ecosystem-based security strategies. Knowing when, and where strategies should be employed and their potential impact in reducing the level of disturbance to network services will contribute to their effective usage. The link to security disturbances are considered through the exploitation route, the attack method, and the attack motivation. Understanding this linkage can help to develop strategies that target vulnerabilities associated with a particular disturbance event such as malware propagation.

1. Introduction

Computer networks and their application environments can be thought of as digital ecosystems [1, 2] where social and technical interactions between devices, users, organisations, and the physical environment combine to produce a multitude of services (see figure 1).

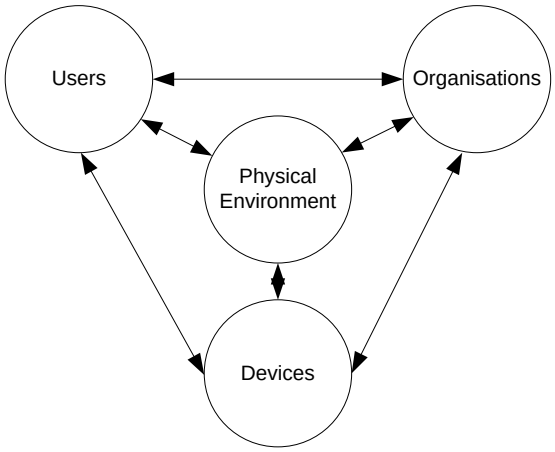

Figure 1 - Digital ecosystem

Many security techniques for computer networks are focused upon detection and removal of known threats, but unknown attacks can happen very quickly creating wide spread devastation. New techniques are therefore required which enable networks to be more tolerant of malicious activity, even when there is an entirely new form of attack. By considering such networks from an ecosystem perspective facilitates the adoption of ecological principles into security policies. Within such a paradigm, security attacks can be thought of as destructive disturbances at the individual, community, or ecosystem scale affecting the functioning of the services provided by the network. Some previous work has looked at how natural ecosystems tolerate disturbances such as viruses and droughts, and how these underlying principles such as biodiversity can be applied to improve the security of computer networks [3]. The research proposed an ecosystem resilience model for a computer network, hypothesising that

the destructive effect on function and services from security attack disturbances can be counterbalanced by constructive biodiversity strategies.

In order to improve digital ecosystem tolerance to disturbance and devise effective biodiversity strategies it is necessary to understand the vulnerabilities that lie at all scales within a digital ecosystem and their link to particular disturbances. The research here investigates the location of vulnerabilities at the individual, community and ecosystem scales and links these to the disturbances that arise within such an environment. This location analysis will provide some insight into where vulnerabilities are located and which aspects could be made more diverse.

2. Vulnerabilities

When assessing a computer network and it's environment from an ecosystem perspective, it is necessary to consider vulnerabilities arising from; the hardware and the software at the individual scale, the interactions at the community scale, and the beneficial users, policy makers and organisations at the ecosystem scale. All of these need to be considered simultaneously because diversity will need to be employed at all scales for maximum effectiveness. There is a danger that if this is not achieved correctly a shift in exploitation may occur, for example from software malware to hardware malware [4].

There has been a great deal of research into security attacks and their classifications. There has been some, but less research into vulnerabilities and their classifications, and even less research into general links between vulnerabilities and attacks. This may be partly due to companies' reluctance in publicising information regarding weaknesses in their products. It may also be because new vulnerabilities and exploits are being discovered every day, usually by specialist researchers or hackers and are often discussed on a case by case basis. Because of this, the scope associated with vulnerabilities is very large. Published research of multiple vulnerabilities are often limited to specific systems, for example Kamara et all [5] did an analysis of vulnerabilities in internet firewalls, and Jiwani and Zelkowitz [6] looked at security flaws in operating systems. Whilst these are useful for the specialised areas they do not cover the scope associated with a digital ecosystem. There are a number of public databases and lists that exist containing vulnerability information, such as the National Vulnerability Database (NVD) [7], the Open Source Vulnerability Database (OSVDB) [8], the Common Vulnerabilities and Exposures (CVE) list [9], and The Common Weakness Enumeration (CWE) dictionary of software weakness types [10]. Although they contain a lot of information, they are mostly software focused and are far from user friendly. When this study was conducted entries were often sparse or incomplete and generating queries for specific information or gathering statistical data was difficult. The Open Web Application Security Project (OWASP) [11] is more user friendly but limits the scope to vulnerabilities and attacks of web applications only. Despite this limitation, many of the web-based vulnerabilities feature highly in other databases as important general software vulnerabilities that should be avoided. This is not surprising since we are becoming more and more dominated by web applications and services.

This paper brings together a number of sources of information to highlight some of the most popular vulnerabilities that could affect digital ecosystems at different scales. The location of the vulnerability is considered from a time (source), physical, and functional perspective; all aspects which are important within ecosystems and defining biodiversity strategies. The links between vulnerability and attack disturbances is investigated by looking at the effects of the vulnerabilities, the attack methods used to exploit the vulnerability, and the motivations of the attack.

3. Vulnerability Location

The 'location' of a vulnerability can mean different things depending upon the research focus. Jiwnani and Zelkowitz [6] use vulnerability location for software auditing to mean functional areas. Garcia et all [12] uses vulnerability location to mean physical layers of operating system code. The CWE

list uses the time of introduction as the source location. Previous research [3] suggests that all three of these aspects: physical (or spatial), functional, and time (or source) are important parameters in the biodiversity of healthy ecosystems. Whilst source location and functional location can be generalised into areas that are transparent across scales, physical location is more specific across scales because is dependent upon the architecture and physical attributes. A list of these areas is given in table 1.

Table 1 – Vulnerability Location

Source	Functional	Physical
Design Implementation\ Manufacture Configuration\ Installation Operation	Monitoring Resource Management Authorisation\ Authentication Data Processing Input\Output Interaction	Specific to each scale – see Table 2

Source Location

The categories for source location were chosen because they can be generalised across scales. They adopt those used within CWE, with the addition of 'manufacture' and 'installation' to make it clear how hardware can be incorporated, as well as software. Vulnerabilities introduced at the 'design' stage can be from poor design decisions, inadequate requirements, or the adoption of weak security policies. Vulnerabilities introduced at the 'implementation/manufacture' stage can be from poor coding practices, inadequate code checking tools or processes, or the use of programming languages that are prone to specific vulnerabilities such as buffer overflows. Not all vulnerabilities are accidental however; they may be inserted intentionally, for example loop holes to allow product manufactures back door entries into devices, foreign government spying, and insiders with a malicious intent or motivated by financial gain. Vulnerabilities introduced at the 'configuration/installation' stage can be based on whether the final product is configured or installed as intended, for example ensuring the appropriate security mechanisms have been set-up. Vulnerabilities introduced at the 'operation' stage could stem from user interaction, incorrect use, or environmental effects.

Functional Location

The majority of vulnerability research is focused upon software as this is where the majority of vulnerabilities currently lie. Jiwnani and Zelkowitz [6] use functional location as part of their susceptibility matrix for categorising operating system flaws. Categories include system initialization, memory management, process management, device management, file management, and identification/authentication. Kamara et all [5] uses functional operations for analysing vulnerabilities within firewalls including application level, reassembly, IP/port filtering, legality checks, dynamic rule set, and NAT/PAT. The categories used within this study were chosen to be transparent across scales in order to generate a coherent structure. They were devised by generalising the scale specific categories, such as those listed above. The 'Monitoring' category includes keeping track of device location, network monitoring, and software and hardware behaviour monitoring. 'Resource management' includes memory, process, device, and file management at the lower scales, as well as resource management and topology control in terms of the whole network community at the upper scales. 'Authorisation\Authentication' includes not only these two access control security features but also covers other features such as validating access, identification and encryption and in some cases is closely linked to input/output interaction which could include messages and commands or user interaction for example. Data processing' covers the processing of signals or data in the hardware, data processing and algorithms in the software, routing of network data messages within the community, and data processing services within the ecosystem. 'Input\Output Interaction' includes the interaction between hardware components and the input and output to the hardware, the interaction between software components and the input and output to the software, the interaction between nodes and the input and output to the

community, and the interaction between communities and other ecosystem components as well as the input and output to the ecosystem.

Physical Location

The physical location of vulnerabilities has been segregated into physical partitions or layers, as shown in table 2, which are different at each scale. Where possible these partitions have been adopted from the literature. Where this was not possible, partitions have been chosen to naturally reflect the vulnerabilities listed in the matrices, section 11. The individual scale is split between hardware and software, each with a different list of physical locations. The hardware locations include the primary components associated with modern printed circuit boards. These include integrated circuits, embedded software, firmware, and inputs and outputs. Integrated circuits include the processor chip, memory, and driver chips etc. Some of the software locations have been adopted from Garcia et all's research on operating system vulnerabilities [12], where OS Drivers, and OS System Software are used. Extra categories have been created to separate vulnerabilities within applications such as web applications, database applications, and other applications. Vulnerabilities within the community scale reflect those associated with network operations rather than the actual software vulnerabilities which may be present. This is reflected in table 2 where the community partitions are located on the right hand side spanning hardware and software partitions on the left. Categories at the community scale have been largely adopted from Wood and Stankovic's [13] categorisation of attacks since many vulnerabilities at this scale lie within the communication mechanism between devices where those based on wireless connections are particularly vulnerable. They use a subset of the OSI model to describe specific attack methods and their vulnerabilities at each layer of the network. Categories include physical, link, network, and transport. The application category has been added here to reflect additional application level vulnerabilities in the network. Ecosystem partitions include all the physical nodes, the users, government and commercial organisations, and the application environment.

Table 2 – Physical Location Partitions

<table>
<tr><td colspan="4">Application Environment</td></tr>
<tr><td colspan="2">Users</td><td colspan="2">Government/Commercial Organisation</td></tr>
<tr><td colspan="4">All Physical Nodes</td></tr>
<tr><td>Web App</td><td>Database App</td><td>Other App</td><td rowspan="2">Application</td></tr>
<tr><td colspan="3">OS System Software & services</td></tr>
<tr><td colspan="3" rowspan="2">OS Core libraries</td><td>Transport</td></tr>
<tr><td>Network</td></tr>
<tr><td colspan="3" rowspan="2">OS Drivers</td><td>Link</td></tr>
<tr><td rowspan="4">Physical</td></tr>
<tr><td>Embedded Software</td><td colspan="2">Firmware</td></tr>
<tr><td colspan="3">Integrated Circuits</td></tr>
<tr><td colspan="3">Inputs and Outputs</td></tr>
</table>

4. Vulnerability – Disturbance

Within a digital ecosystem security attacks are considered as disturbances and vulnerabilities allow these to take place. The vulnerability itself will determine the type of 'exploitation route(s)' that the disturbance may use. How the disturbance is achieved is determined by the 'attack method', and the attack method is often incited by an 'attack motivation'.

Exploitation routes and attack motivations have been generalised into categories that are transparent across scales, although not all are applicable to every scale. A list of these are given in table 3. Attack methods are specific for each scale because they are dependent upon the architecture and physical characteristics. These attack methods are listed in tables 5, 7, 9, and 11.

Table 3 – Vulnerability Disturbance

Exploitation route	**Attack method**	**Attack motivation**
By-pass protection mechanism Information leakage Tampering Social Engineering Execute code/command Denial of service	Specific to each scale see Tables 5,7,9,11	Service Disruption Identity Theft Data Theft Malicious Control

Exploitation route

When considering the linkage between vulnerabilities and disturbances it is useful to identify the effect that the vulnerability may have. For example the vulnerability may allow security mechanisms to be by-passed, or the execution of code. This gives the attacker a route for exploitation. For software vulnerabilities, CWE lists these as 'effects' and includes: by-pass protection mechanism, read application data, modify application data, read files, modify files, execute unauthorised code/command, denial of service - crash/ exit/restart, and denial of service – resource consumption. In terms of hardware, Grand [14] defines a list of threat types: interception, interruption, modification style, and fabrication style, whereas Li et all [15] defines hardware attack categories as: information leakage, tampering, and denial of service. Although the literature describes them as threat or attack types they can be considered as general categories for exploitation routes. Denial of service for example is categorised as a software vulnerability effect, and file\data modification could be considered as modification style or tampering. At the community scale a majority of the vulnerabilities permit a denial of service as the exploitation route. At the ecosystem scale social engineering and security loopholes provide the necessary exploitation paths. Combining these into a general list of exploitation routes is given in table 3.

Attack Methods

Attack methods listed within this paper are specific for each scale because they are dependent upon the architecture. As stated previously there has been much literature describing comprehensive taxonomies for attacks, but there has been little research in trying to formulate links between vulnerabilities and attack methods. Most of the attack methods listed here, and their links to vulnerabilities, have been inferred from the attack and vulnerability literature which is discussed further in sections 5 to 8. The CWE list of vulnerabilities however, attempts to bridge the gap for software. Each vulnerability description lists relevant attack patterns that cross reference to a CAPEC (Common Attack Pattern Enumeration and Classification) number. CAPEC is a public dictionary of security attack methods for software [16]. Because the CAPEC dictionary is hierarchical it can also be viewed as a method of classification, although CWE does not distinguish this hierarchy within the referencing. This cross referencing of CAPEC has therefore been adopted for the software attack methods listed within this paper (Table 7).

Attack Motivation

The motivation of an attacker is relevent to every scale of an ecosystem, but the 'details' about what motivates an attacker is not widely researched. Rounds and Pendgraft [17] suggest that attacker motivations are linked to fun and adventure, recognition, political, to investigate criminal activity, state sponsored, and financial and defensive. Gandhi et all [18] described that cyber attack motives fit into three categories: political factors, socio-cultural factors and economic factors. Grand [14] defines motivation in terms of attack goals for hardware which include: competition, theft of service, user authentication, privilege escalation, focused attack, lunchtime attack style, insider attack. Li et all [15] stated that most attack goals, which can also be attack types, can be classified into three categories; information leakage, tampering, and denial of service. The categories defined within this study use attack goals as the motivating factors, and have been adapted from those listed above to include the following categories: service disruption, identity theft, data theft, and malicious control. Service disruption signifies that the motive is to disrupt services at the node, the network, or the ecosystem. Identity theft can often mean impersonation at the node level or stealing a user's identity at the ecosystem level for acts such as credit card fraud. Data theft covers a wide range of motives, from stealing data for financial gain, to exposing sensitive information. Malicious control signifies the motivation for the control of nodes such as zombies for botnets.

5. Individual (Hardware) Vulnerability matrix

In the past, the focus of device security has been on software, but it is not just software that is

vulnerable to security attacks. The vulnerability matrices (Tables 4 and 5) include hardware vulnerabilities that are being discussed in the media and research. One growing concern is the methods by which chips are designed and manufactured. Global outsourcing has become more common in recent years due to the increased complexity of chip designs whilst minimising costs. Many people and organisations responsible for the design and manufacture of such chips are spread around the globe. It is entirely possible for hardware Trojans to be inserted secretly within a computer chip due to the outsourcing of the design of sub-sections of the chip or buying in third party IP [19]. The Trojans could be placed at any functional location within the hardware, and can be physically placed within the circuitry of a dedicated ASIC (Application Specific Integrated Circuit), implemented within the firmware of reconfigurable devices, or hidden within embedded software pre-programmed into memory. Given that it is not possible to test every input condition, such Trojans are likely to go unnoticed until sometime later when their effects become apparent. In addition to this, many hardware devices are fixed so cannot be patched like software, and therefore need to be replaced. There are some strategies being developed to detect such malicious circuits whilst in operation. They mostly involve including extra hardware to police the chip from the inside [20]. However, there are examples where products have already been shipped with malware such as a password stealing Trojan that infected Seagate disk drives built in China in 2007, as well as Samsung digital picture frames which had to be recalled in 2008 when malware was found pre-loaded on the internal storage [21]. Another hardware vulnerability that is widely documented is the vulnerability of a device's physical inputs and outputs to manipulation or monitoring which is an inherent design feature of an electronic device. Such attacks exploiting this vulnerability include eavesdropping and fault injection which can lead to message corruption. One such example of transient fault injection was demonstrated in 2010 by altering the voltage supply of a programmable device in order to analyse the corrupted messages and extract the system's 1024-bit RSA private key information [22]. Another vulnerability that is often considered as part of the hardware is low level chip code such as BIOS code. But like all software, it can still contain exploitable vulnerabilities. These are not listed here; instead, specific software vulnerabilities are discussed in section 6 and listed in Table 6/7. An example of an attack exploiting BIOS code was in 2008 when an exploit was discovered that could take advantage of an Intel CPU caching vulnerability to gain unauthorised access to a protected region of system memory [23], thereby by-passing security mechanisms and allowing information leakage or modification. This type of vulnerability can stem from design or implementation errors. The last hardware vulnerability that will be considered here is the vulnerability associated with the fact that hardware is a physical object that can be accessed. The source of this could be inherent in the design, or introduced at the installation stage where the device is not physically secured. The effect associated with this is that the hardware can then be tampered with, leading to physical modification, internal damage, or the extraction of data. Motives include service disruption and data theft.

6. Individual (Software) Vulnerability matrix

There are various schemes for categorising software vulnerabilities and attacks [24, 25]. More specific information regarding actual software vulnerabilities and their attributes can be found in vulnerability databases. These are often large, and extracting information regarding the most prevalent vulnerabilities is difficult. A more focused list has been produced as a collaboration between MITRE, the SANS Institute and other security experts. The yearly list details the top 25 most dangerous software errors [26]. The list for 2010 has been used here as the basis for investigating software vulnerabilities, their location, and links to attacks. The list uses CWE identifiers, which in turn reference CAPEC attack patterns (Tables 6 and 7). Interestingly, all of the top 25 software vulnerabilities can be located within web applications, and a large proportion can also be found in database applications. More and more of our lives involve interacting with the

web and generating or retrieving data, so this is therefore not an unsurprising result. More traditional vulnerabilities still exist; for example buffer overflow vulnerabilities are still a major problem and are discussed further.

At the top of this software vulnerability list is the improper Neutralization of inputs during web page generation (no. 1). This vulnerability is located in web applications and is described as the most prevalent, obstinate, and dangerous vulnerability. It is introduced at the design or implementation stage when non-vetted libraries or frameworks are used, and untrusted inputs are not accounted for or mitigated against. This type of vulnerability can lead to protection mechanisms being by-passed, information leaks tampering, and the execution of unauthorised code or commands. The most common attack method is via script injection, often called cross-site-scripting, where attackers can inject Javascript or other content into a web page that the unsuspecting application generates. The web page is then accessed by other users, whose browsers execute the malicious script as if it came from the unsuspecting application. Given the fact that this vulnerability is located in web applications, it is not surprising that cross-site scripting attacks are high on OWASP's top 10 list of the worst security risks in web applications[27].

The second most dangerous software vulnerability is the improper neutralization of special elements used in an SQL command (no. 2). This is often found in web-based databases. As we generate more and more data, so will the necessity of web-based database manipulation; such as submitting and extracting data, and searching and filtering information. If mitigation measures are not taken at the design, implementation and operation stages, attackers can modify SQL queries used in security controls such as authentication to bypass security and steal, corrupt, or modify data. This type of attack is called SQL injection and is listed as the number one web application security risk in 2010 by OWASP. Other vulnerabilities that do not properly deal with external inputs include the reliance on untrusted inputs in a security decision (no. 6), improper limitation of a pathname to a restricted directory (no.7), and improper neutralisation of special elements in an OS command (no. 9). An attacker can change these inputs affecting authentication and authorisation allowing them to by-pass security. Another related vulnerability, but is unique to a single programming language is the improper control of filename for include/require statement in PHP program (no. 13). This is where the PHP application incorrectly restricts the input before its usage in 'require' and 'include' functions. This can allow an attacker to execute code by specifying a URL to a remote location or a local file.

The third vulnerability is decades old, but despite this, is still in the top three. A buffer copy without checking the size of the input often leads to the well known classic buffer overflow attack. This vulnerability can be located in any part of the code or functionality, but stems from implementation errors when using programming languages such as C or C++. As a result it is not usually found in web applications which use other programming languages such as Java, but the vulnerability is still prominent in general and underlying software. This vulnerability can allow the execution of unauthorised code or commands, or allow a denial of service through a crash or resource consumption. Other buffer or memory array related vulnerabilities include buffer access with incorrect length value (no.12), improper validation of array index (no.14), integer overflow or wraparound (no.17), and incorrect calculation of buffer size (no.18).

MITRE and SANS call the fourth vulnerability cross-site request forgery which is often given as the name of the corresponding attack method. This vulnerability lies within the web application and comes fifth on OWASP's risk list. Here, the application does not sufficiently verify whether a request was intentionally provided by the user who submitted the request. In this scenario it is possible for an attacker to trick a user into making an unintentional request to the web server which is then treated as an authentic request. This can lead to data theft or service disruption.

Improper authorisation is fifth on the list. This vulnerability is due to the software not correctly authorising actions such as accessing resources. If users are able to access data or perform actions

that are not allowed, this can lead to security mechanisms being bypassed, information leakage, denial of service, and code execution. Another vulnerability related to the access of resources is the incorrect permission assignment for critical resource (no.21) where a resource is given a permissions setting providing access to a wider range than required. This can lead to the exposure of sensitive information. When the software allocates a resource without imposing any restrictions on how many resources can be allocated (no. 22), an attacker can prevent others from accessing the same type of resource. Missing authentication for critical function (no. 19) is where the software does not perform authentication for functionality that consumes a significant amount of resources or requires a provable user identity. Attacker's access will depend on the associated functionality, but can range from reading or modifying data, access to privileged functionality, or the execution of code.

Vulnerabilities that allow unrestricted upload (no.8) and download (no.20) of files can allow code to be run from a remote location, or allow files with dangerous extensions such as .php to be automatically processed without restrictions.

Some software does not encrypt sensitive data when it should (no. 10), or does not use a proper cryptographic algorithm (no.24). If a secure channel is not used, such as SSL, or a tried and tested cryptographic algorithm to exchange sensitive information, it is possible for an attacker with access to the network traffic to eavesdrop and uncover the data. If a good cryptographic algorithm is used, it can then become ineffective if the key is hard coded into the software (no. 11). This error is surprisingly high up on the vulnerability list and allows an attacker to bypass the authentication that has been configured by the software administrator.

Other unusual vulnerabilities that have made it to the top 25 include Software that does not properly check for unusual conditions that are not expected to occur on a day to day basis (no. 15) can miss loopholes that can be exploited by attackers. It may be assumed that certain conditions will never occur, such as low memory, or no access to resources, but attackers may use these unusual conditions to create instability or incorrect behaviour. Information exposure through an error message (no. 16) can be created by including too much error information such as personally identifiable information, authentication, and server configuration. This data can be used by any attacker to misuse software. A race condition (no. 25) vulnerability can involve multiple processes or multiple threads in which the attacker has control over a process or thread. This can have an impact when the expected synchronisation is in security-critical code, such as during authentication.

7. Community Vulnerability matrix

Vulnerabilities at the community scale are located within the communication and data flow, as well as those associated with physical node location, distribution, and mobility. Vulnerabilities within the network are often defined based upon their enabled attack scenario rather than the location of the actual vulnerability. Wood and Stankovic [13] however, categorise attacks into physical layers of the network stack. Bicakci and Tavli [28] discuss network attacks at the physical and MAC layers. Mishra & Nadkarni [29], Lane [30] and Wu et all [31] discuss attacks and vulnerabilities in wireless networks. This literature has been used to establish the links between vulnerability and location, and vulnerability and disturbance at the community scale. The detailed descriptions of each attack technique allow the various vulnerabilities and their links to be inferred (Tables 8 and 9).

Mobile and wireless nodes of a network are inherently more vulnerable than static ones (no.3). For example, there is no guarantee that a path between two nodes would be free of malicious nodes not complying with the employed protocol. This could lead to many attacks such as impersonation, repudiation, or the modification of routing messages such as black holes. Another vulnerability associated with the physical attributes of a community is the knowledge of the location of a device or community (no.1) such as a corporate network, or important building, particularly if they are physically insecure. This could allow targeted attacks such as the

introduction of a virus directed at a specific network location. Other techniques can be used such as impersonation or eavesdropping, or even the simple theft of devices. This is different from the individual scale where vulnerability was concerned with access 'inside' the device. Wireless networks that use fixed frequencies (no. 2) at the physical layer can be subject to jamming attacks. Frequency hopping is used to avoid this, but can still be subject to wide band jammers. For this reason some protocols also use spread spectrum, and code spreading techniques. This vulnerability is likely to be more prevalent in sensor networks where low power requirements force the design of fixed frequency communications.

At the link layer, a number of vulnerabilities allow attackers to cause service disruption. These include: the transmissions from one node to another can be detected (no. 4) giving an attacker the opportunity to induce collisions, protocols that allow unlimited transmission requests (no.5) can be subject to resource exhaustion, channel priority schemes that rely on co-operation (no.6) may be unfairly blocked channel access, and protocols that rely on MAC address filtering (no.7) to identify individuals may have addresses deliberately changed for impersonation attacks.

The network layer is concerned with routing, and within a mobile network every node is potentially a router with associated vulnerabilities. The use of dynamic routing (no. 8) can lead to misdirection or neglect and greed, where messages are incorrectly routed, not routed or they are given priority to the malicious node's own messages. Using multiple (diverse) routing paths or sending redundant messages can reduce the effect of this attack. Location-based protocols that rely on geographic forwarding (no. 9) can allow homing attacks where a malicious node monitors routing traffic, and once the critical resources are found they can be attacked using other methods. Distance vector based protocols allow inconsistent route advertisements (no. 10) across the network meaning zero cost routes can be formed creating black hole attacks. On the other hand stateless protocols without source authentication (no.12) can allow incorrect replies disrupting the node. Networks that do not employ some kind of monitoring (no.11) scheme such as intrusion detection can make it difficult to detect when, how and what attacks are taking place, allowing all kinds of unknown malicious activity to continue un-noticed. Wireless networks tend to have a lower bandwidth limitation (no. 13) than wired networks and therefore attacks on wireless networks from a wired network can easily create flooding attacks.

At the transport layer, protocols that must maintain state in memory (no. 14) are vulnerable to memory exhaustion through flooding where each request causes the victim to allocate memory for that particular connection. Some protocols at the transport layer allow synchronisation recovery (no.15) to retransmit missed frames. If the malicious node can maintain timing, it can prevent the end points from exchanging information, causing them to enter into an endless synchronisation-recovery cycle. When protocols authenticate at session set up (no.16), but not thereafter, a session hijacking attack may occur where the attacker steals the victim's identity and then continues the session with the target node. This is known to occur in TCP and UDP protocols for example.

Security techniques can be used to inspect or encrypt data but the decision to actually participate in communication (no.17) is down to the individual node. At the application layer this can be seen as a vulnerability where repudiation attacks can occur in which a node does not participate in all or part of the communications mechanism. Also at the application layer many user data types and protocols are supported (no.18) allowing malware to hitch a ride. Although malware can propagate at different layers of the network, the majority use the application layer. It is the act of connecting nodes together that facilitates the movement of such malware attack methods. The vulnerability effect of employing no or weak security can cause security mechanisms to be by-passed (no.19). This vulnerability can originate at the design stage or from poor configuration. It can be located at any of the physical layers but is only applicable to the functional location of authorisation / authentication. At the physical network layer this

may mean using scrambling, hopping, or spread spectrum techniques, at the link layer this may mean the use of encryption such as WEP (WLAN), A5 (GSM), and WPA, at the network layer this may mean using IPsec, at the transport layer this may mean using secure protocols such as SSL, TLS, and WTLS, at the application layer this may mean using ssh, S/MIME, and PGP.

8. Ecosystem Vulnerability matrix

There is little information on the specific analysis of security related vulnerabilities at the ecosystem scale within a digital ecosystem as defined within the context of this research. A small number of publications and reports have been reviewed both from the ecology domain and the digital security domain and have been adapted to fit into the context of this study. Digital ecosystems include not only communities of nodes working together to provide services, but also the physical application environment, users, organisations and regulations. Vulnerabilities have been included that reflect these areas (Tables 10 and 11).

Users at the ecosystem scale who utilise the devices within the network and benefit from the services they provide have a number of universal vulnerabilities which can create common exploitation routes for cyber criminals [32]. The natural human user tendency to trust (no.1) is often exploited through social engineering. Attacks such as phishing, malicious emails and malware can often be motivated by identity and data theft, normally for financial purposes. Innocent users posting personal information on web sites or chat rooms, or even leaving company data on memory sticks (no.2) can accidentally leak information which can be used later by attackers. Often users lack technical skills (no.3) which can give rise to security mechanisms being by-passed, especially if they are not adequately set up or activated. Users also often have short or easy to crack passwords (no.5), which can result in security mechanisms being easily by-passed. Finally, depending upon the device, users may fail to monitor their device regularly (no.6) or simply forget where it was last located (no.4). Discovery by a malicious attacker could lead to Identity or data theft.

Zavaleta et all [33] discusses vulnerabilities in natural ecosystems including those associated with whole communities. There are two relevant vulnerabilities to digital ecosystems. The first is that communities that are responsible for the generation, storage or transport of valued information may be more vulnerable to attacks (no.7). These attacks can appear at the individual or community scales in the form of targeted attacks exploiting software and hardware vulnerabilities, or causing network service disruptions. The second concerns communities that operate within a particular application environment that is either hostile or targeted by a particular threat (no.8). Such scenarios will be more vulnerable, and examples could involve a war zone where the risk from destructive attacks may be greater. Within an ecosystem there may be scenarios where device operations and the application is sensitive to environmental effects (no.9). So for example a body sensor network primarily used in the dry under a coat, suddenly becomes exposed to rain may have an effect on the sensor readings, processing of the data, and information being fed back to the main monitoring device. In this case this particular ecosystem is vulnerable to large environmental changes outside its normal operating range. An attacker may leverage this kind of vulnerability to disrupt nodes or falsify data. Weak policies on security standards, network operations and construction (no.10) is a vulnerability at the ecosystem scale allowing attackers to by-pass security and gain unauthorised access more easily. This type of vulnerability is filtered down to lower scales. The lack of adoption of new technology or upgrades (no.11) by users and manufacturers is another vulnerability giving attackers an easier route for entry. Systems that are not up to date expose many lower-scale vulnerabilities. Devices that are used for an application for which it was not intended (no.12) can lead to the exposure of untested bugs and vulnerabilities that cannot be patched. Jailbreaked i-phones for example allow users to install non-Apple applications, but have also been the target of malware Trojans.

Another vulnerability includes the lack of general diversity (no.13) amongst communities and between communities which can lead to the same

vulnerability being targeted across many devices leading to the propagation of malware for example [34].

The physical layout of the ecosystem (no.14), or connected topology may make it more vulnerable than others. Some topologies may be less resilient when faced with an attack scenario. In a server-client network topology for example, if the server is compromised it could affect the entire network. In a distributed peer to peer network there is no main server so compromised devices can be removed without affecting the whole network. In natural ecosystems a study [35] found that fragmentation, where communities were spread out and their locations were sparsely populated, resulted in ecosystems less able to adapt to disturbances making them less resilient.

9. Discussion

The vulnerability matrices allow consideration of vulnerabilities and their link to disturbances from different perspectives. This may assist in understanding when and where biodiversity strategies should be employed to mitigate against a specific vulnerability or disturbance.

For example suppose it is required to address the hardware vulnerability of chip design complexity that requires outsourced manufacturing resulting in possible hidden malware or malicious circuits. It would be inappropriate to just concentrate on the manufacturing process of chips, because the design process also needs to be considered. In addition, consideration is needed towards the embedded software and firmware running on the chip across most functional categories. Any unaddressed aspect of the vulnerability will create a loop hole and an alternative route of exploitation.

If instead the requirement is to stop the spread of malware, then vulnerabilities that allow the execution of code/commands and specific malware attacks need to be addressed. This covers a large number of vulnerabilities at all scales from low level chip code, to buffer copy software errors to communities of nodes supporting different data types. Such a requirement would include the vulnerabilities given in table 12.

Table 12 – Vulnerabilities associated with malware spread

		Vulnerability ID
Ecosystem		1,3,6,10,11,12,13,14
Community		1,11,18,19
Individual	**SW**	1-5, 7-9, 11-15, 17,18,20,21,23,25
	HW	1,3

The incompleteness of the literature did not enable vulnerabilities outside of software to be listed in order of importance. Also the literature did not allow the inclusion of more specific attack methods outside of software. The information contained here has been extracted from the literature, or where this was not possible, inferred from the literature and as such some aspects may be open to interpretation. Even allowing for this, an observation regarding software vulnerabilities, in comparison to other vulnerabilities, is that they are more likely to be located in more than one physical location. There are some vulnerabilities that are found within web applications only, but vulnerabilities associated with buffers for example could potentially exist in many physical locations. In these instances the vulnerabilities seem to be better separable through functional location. This may be an important consideration for mitigation strategies.

10. Conclusion

This study considers computer networks from an ecosystem perspective with a focus on the location of vulnerabilities at different scales and their link to undesirable security disturbances. The location of vulnerabilities are considered from a physical, functional and time perspective. The link to security disturbances are considered through the exploitation route, the attack method, and the attack motivation. The matrices allow a specific vulnerability or disturbance to be analysed. For example there are many vulnerabilities associated with the spreading of malware across all scales. Buffer related vulnerabilities for example could potentially exist in many physical locations. In these instances the vulnerabilities seem to be better separable through functional location.

10. Acknowledgements

This work was supported in part by the Complexity Science Doctoral Training Centre at the University of Warwick under EPSRC funding. This document was first created in October 2011, and published in Warwick's archives (WRAP) in February 2017.

11. Vulnerability Matrices

Table 4 – Individual (Hardware) Vulnerability-Location Matrix

	Location												
	Physical				Functional					Source			
Vulnerability	Integrated circuit	Firmware	Embedded software	Inputs and outputs	Monitoring	Resource Management	Authorisation/Authentication	Data Processing	Input/ Output Interaction	Design	Implementation/manufacture	Configuration/installation	Operation
1 Chip design complexity requires third party IP and outsource manufacturing	p	p	p		f	f	f	f		s	s		
2 Hardware inputs and outputs are vulnerable to manipulation or monitoring				p					f	s			
3 Low level chip code vulnerability (see software vulnerability list Table 6/7, non web-based, non database-based)			p		f	f	f	f	f	s	s		
4 Hardware physical access inside device				p					f	s		s	

Table 5 – Individual (Hardware) Vulnerability-Disturbance Matrix

	Disturbance																
	Exploitation Route						Attack Methods							Attack Goals			
Vulnerability	By-pass security mechanism	Information Leakage	Tampering	Social engineering	Execute code\command	Denial of service	Hidden malicious circuits (Trojan)	Hidden malware (Trojans)	Transient Fault Injection	Eavesdropping	Software attacks (see Table 7)	Physical modification/ damage	Data / memory theft	Service disruption	Identity theft	Data Theft	Under malicious control
1 Chip design complexity requires third party IP and outsource manufacturing			r				m	m						g		g	g
2 Hardware inputs and outputs are vulnerable to manipulation or monitoring		r	r						m	m				g	g	g	
3 Low level chip code vulnerability (see software vulnerability list Table 6/7, non web-based, non database-based)	r	r	r		r	r		m			m			g		g	g
4 Hardware physical access inside device			r									m	m	g		g	

Table 6 – Individual (Software) Vulnerability-Location Matrix

	Location														
	Physical						Functional					Source			
Vulnerability	OS Drivers	OS Core libraries	OS System software	Web applications	Database applications	Other Applications	Monitoring	Resource Management	Authorisation /Authen.	Data processing	Input/ Output	Design	Implementation/man.	Configuration/installati	Operation
1 Improper Neutralization of Input During Web Page Generation				p							f	s	s		
2 Improper Neutralization of Special Elements used in an SQL Command				p	p				f		f	s	s		s
3 Buffer Copy without Checking Size of Input	p	p	p	p	p	p		f					s		
4 Cross-Site Request Forgery				P					f		f		s		
5 Improper Authorisation				p	p			f	f			s	s		s
6 Reliance on Untrusted Inputs in a Security Decision				p	p				f		f	s	s		
7 Improper Limitation of a Pathname to a Restricted Directory				p		p		f			f	s	s		
8 Unrestricted Upload of File with Dangerous Type				p							f	s	s		
9 Improper Neutralization of Special Elements used in an OS Command				p							f	s	s		
10 Missing Encryption of Sensitive Data				p	p				f			s			s
11 Use of Hard-coded Credentials	p	p	p	p	p	p			f			s			
12 Buffer Access with Incorrect Length Value	p	p	p	p	p	p		f					s		
13 Improper Control of Filename for Include/Require Statement in PHP Program				p				f			f	s	s		
14 Improper validation of array index	p	p	p	p	p	p		f		f	f		s		
15 Improper Check for Unusual or Exceptional Conditions	p	p	p	p	p	p					f		s		
16 Information Exposure Through an Error Message	p	p	p	p	p	p					f	s	s	s	s
17 Integer Overflow or Wraparound	p	p	p	p	p	p		f		f			s		
18 Incorrect Calculation of Buffer Size	p	p	p	p	p	p		f		f			s		
19 Missing Authentication for Critical Function				p	p	p			f			s			
20 Download of Code Without Integrity Check			p	p					f			s	s		
21 Incorrect Permission Assignment for Critical Resource	p	p	p	p	p	p		f	f			s	s	s	s
22 Allocation of Resources Without Limits or Throttling	p	p	p	p	p	p		f				s	s	s	s
23 URL Redirection to Untrusted Site				p							f	s	s		
24 Use of a Broken or Risky Cryptographic Algorithm		p	p	p		p			f			s			
25 Race Condition			p	p	p	p		f				s	s		

Table 7 – Individual (Software) Vulnerability-Disturbance Matrix

	Disturbances																																	
	Exploitation Routes						Attack Methods (CAPEC number)																										Attack Goals			
Vulnerability	Bypass protection mech.	Information leakage	Tampering	Social Engineering	Execute code\command	Denial of service	Script Injection 242	Exploit privilege/trust 232	SQL injection 66	Overflow buffers 100	Integer attacks 128	Cross Site Req. Forgery 62	Data excavation 116	Exploit authentication 225	Probabilistic tech. 223	Registry manipulation 269	File manipulation 165	Input manipulation 153	Path traversal 126	Parameter injection 137	command injection 248	Analytic attacks 281	Data interception 117	Character injection 249	Abuse of functionality 210	Impersonation 151	Remote code inclusion 253	Leverage Race cond. 26	Content spoofing 148	Resource Depletion 119	Service disruption	Identity theft	Data theft	Under malicious control
1 Improper Neutralization of Input During Web Page Generation	r	r			r		m	m																					g		g	g
2 Improper Neutralization of Special Elements used in an SQL Cmd.	r	r	r		r				m												m									g	g	
3 Buffer Copy without Checking Size of Input			r		r	r				m	m																		g			g
4 Cross-Site Request Forgery	r	r	r		r	r						m	m																g	g	g	g
5 Improper Access Control	r	r	r		r			m		m				m	m	m														g	g	
6 Reliance on Untrusted Inputs in a Security Decision	r							m																					g		g	
7 Improper Limitation of a Pathname to a Restricted Directory		r	r		r	r		m									m	m	m										g		g	g
8 Unrestricted Upload of File with Dangerous Type					r			m																					g			g
9 Improper Neutralization of Special Elements used in an OS Command		r	r		r	r			m									m		m	m								g		g	g
10 Missing Encryption of Sensitive Data		r	r					m							m							m			m					g	g	
11 Use of Hard-coded Credentials	r	r			r			m							m						m								g	g	g	g
12 Buffer Access with Incorrect Length Value			r		r	r				m																			g			g
13 Improper Control of Filename for Include/Require in PHP Program			r		r					m																m			g			g
14 Improper validation of array index		r	r		r	r				m																			g		g	g
15 Improper Check for Unusual or Exceptional Conditions	r		r		r	r																		m					g			g
16 Information Exposure Through an Error Message		r	r						m				m											m							g	
17 Integer Overflow or Wraparound			r		r	r					m																		g			g
18 Incorrect Calculation of Buffer Size			r		r	r				m																			g			g
19 Missing Authentication for Critical Function	r											m		m									m	m						g	g	
20 Download of Code Without Integrity Check			r		r																			m					g			g
21 Incorrect Permission Assignment for Critical Resource		r	r		r			m						m															g	g	g	g
22 Allocation of Resources Without Limits or Throttling						r																		m				m	g			
23 URL Redirection to Untrusted Site	r		r		r	r																					m		g	g		g
24 Use of a Broken or Risky Cryptographic Algorithm	r	r	r												m																g	
25 Race Condition		r	r		r	r																				m			g		g	g

Table 8 – Community Vulnerability-Location Matrix

	Location													
	Physical					Functional					Source			
Vulnerability	Physical	Link	Network	Transport	Application	Monitoring	Resource Management	Authorisation/Authenticati	Data Processing	Input/ Output Interaction	Design	Implementation	Configuration	Operation
1. Physical device or community location	p						f				s			s
2. known/fixed wireless transmission frequency	p									f	s			
3. Dynamic nature of devices	p									f	s			s
4. Transmission can be detected		p								f	s			
5. Supports unlimited transmission requests		p								f	s			
6. Channel priority scheme relies on co-operation		p								f	s			
7. Reliance on MAC address filtering		p								f	s			
8. Individual dynamic routing control			p						f		s			
9. Reliance on geographic forwarding			p						f		s			
10. Inconsistent route advertisements			p						f		s			
11. Lack of network monitoring			p			f					s			
12. Stateless protocol without source authentication			p					f			s			
13. Wireless bandwidth limitations			p							f	s			
14. Protocols that maintain state in memory				p						f	s			
15. Synchronisation recovery without authentication				p				f		f	s			
16. Authentication at session set up only				p				f			s			
17. Node decision to participate in communications					P					f				
18. Connecting nodes supporting different data types					P					f	s			
19. No/weak security – does this need splitting out?	p	p	p	p	p			f			s		s	

Table 9 – Community Vulnerability-Disturbance Matrix

	Disturbance																											
	Exploitation Route						Attack Methods																			Attack Goals			
Vulnerability	By-pass security mechanism	Information Leakage	Tampering	Social engineering	Execute code\command	Denial of service	Jamming	Eavesdropping	Physical modification	Impersonation	Collision	Exhaustion	Unfairness	Neglect & Greed	Homing	Misdirection	Black holes	Incorrect reply	Flooding	De-synchronisation	Repudiation	Session hijacking	Malware propagation	Destruction\theft	Service disruption	Identity theft	Data Theft	Under malicious control
1. Physical device or community location			r						m	m													m	m	g	g	g	g
2. known/fixed wireless transmission frequency		r				r	m	m																	g		g	
3. Dynamic nature of devices	r	r				r				m					m	m	m	m			m				g	g	g	
4. Transmission can be detected						r					m														g			
5. Unlimited transmission requests						r						m													g			
6. Channel priority scheme relies on co-operation						r							m												g			
7. Reliance on MAC address filtering						r				m															g	g	g	
8. Individual dynamic routing control						r								m		m									g		g	
9. Reliance on geographic forwarding						r									m										g		g	
10. Inconsistent route advertisements						r											m								g		g	
11. Lack of network monitoring	r						m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	g	g	g	g
12. Stateless protocol without source authentication						r												m							g			
13. Wireless bandwidth limitations						r													m						g			
14. Protocols that maintain state in memory						r													m						g			
15. Synchronisation recovery without authentication						r														m					g			
16. Authentication at session set up only	r	r				r																m			g	g	g	
17. Node decision to participate in communications																					m				g			
18. Connecting nodes supporting different data types					r																		m		g	g	g	g
19. No/weak security	r							m		m												m	m		g	g	g	g

Table 10 – Ecosystem Vulnerability-Location Matrix

	Location												
	Physical				Functional					Source			
Vulnerability	Users	Government \commercial	Community of nodes	Application environment	Monitoring	Resource Management	Authorisation/Authentication	Data Processing	Input/ Output Interaction	Design (inc policy)	Implementation	Configuration	Operation
1 Natural human user tendency to trust	p								f				s
2 User posting personal information on web sites/chat rooms, leaving company files on memory sticks	p							f	f				s
3 Lack of understanding/computer skills to use the device securely	p						f		f			s	s
4 User does not know or keep track of device location	p				f								s
5 Short, or easy to crack passwords	p						f			s		s	s
6 User fails to monitor device regularly if in remote location such as a sensor network	p				f					s		s	s
7 Community responsible for the generation, storage or transport of valued information.			p					f		s			s
8 Communities of nodes that operate within a particular application environment that is either hostile or targeted by a particular threat.				p					f				s
9 Device operations and application is sensitive to environmental effects			p	p					f	s		s	s
10 Weak policies on security standards, network operations and construction		p				f	f			s			
11 Lack of adoption of new technology or upgrades by users\manufacturers	p	p				f			f	s		s	s
12 Devices used for an application for which it was not intended	p		p	p		f			f				s
13 Low diversity ecosystems			p			f				s			s
14 Poor physical layout of ecosystem or connection topology	p	p	p	p		f				s			s

Table 11 – Ecosystem Vulnerability-Disturbance Matrix

	Disturbance																			
	Exploitation Route						Attack Method										Attack Goal			
Vulnerability	By-pass protection mechanism	Information Leakage	Tampering	Social engineering	Execute code\command	Denial of service	Phishing	Malicious email\ scams	Malicious downloads	Malware propagation	Intelligence gathering	Theft	Environment damage	Software attacks (see table 7)	Hardware attacks (see table 5)	Community attacks (see table 9)	Service disruption	Identity theft	Data theft	Under malicious control
1 Natural human user tendency to trust	r	r		r			m	m	m	m	m							g	g	
2 User posting personal information on web sites/chat rooms, leaving company files on memory sticks		r									m	m						g	g	
3 Lack of understanding/computer skills to use the device securely	r	r					m	m	m	m				m	m		g	g	g	g
4 User does not know or keep track of device location			r									m						g	g	
5 Short, or easy to crack passwords	r													m				g	g	
6 User fails to monitor device regularly if in remote location such as a sensor network		r	r							m	m	m	m	m	m		g	g	g	g
7 Community responsible for the generation, storage or transport of valued information.											m			m	m	m			g	
8 Communities of nodes that operate within a particular application environment that is either hostile or targeted by a particular threat.													m	m	m	m	g	g	g	g
9 Device operations and application is sensitive to environmental effects			r										m				g			
10 Weak policies on security standards, network operations and construction	r						m	m	m	m				m	m	m	g	g	g	g
11 Lack of adoption of new technology or upgrades by users\manufacturers	r									m				m	m		g	g	g	g
12 Devices used for an application for which it was not intended	r		r						m	m				m	m		g	g	g	g
13 Low diversity ecosystems	r	r	r		r	r			m	m	m		m	m	m	m	g		g	g
14 Poor physical layout of ecosystem or connection topology	r	r	r	r	r	r				m				m	m		g		g	g

References

[1] H. Boley, and E. Chang, “Digital ecosystems: Principles and semantics,” in IEEE International Conference on Digital Ecosystems and Technologies, 2007.

[2] T. Yamakami, “A mobile digital ecosystem framework: Lessons from the evolution of mobile data services,” in 13th International Conference on Networked-Based Information Systems, 2010.

[3] J. Jackson, S. creese, and M. S. Leeson, “Biodiversity: A security approach for ad hoc networks,” in IEEE Symposium on Computational Intelligence in Cyber Security, Paris, France, 2011.

[4] K. Jackson-Higgins. "Possible new threat: Malware that targets hardware," Dark Reading, www.darkreading.com.

[5] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, “Analysis of vulnerabilities in internet firewalls,” *Computers and Security,* vol. 22, no. 3, 2003.

[6] K. Jiwnani, and M. Zelkowitz, “Susceptibility matrix: A new aid to software auditing,” *IEEE Security & Privacy,* vol. March/April, 2004.

[7] "National vulnerability database." National Institute of Standards and Technology, http://nvd.nist.gov.

[8] "The open source vulnerability database." http://osvdb.org.

[9] MITRE. "Common vulnerabilities and exposures," MITRE, http://cve.mitre.org, https://www.cvedetails.com.

[10] "Common weakness enumeration, a community developed dictionary of software weakness types." http://cwe.mitre.org/.

[11] "Owasp the open web application security project." www.owasp.org.

[12] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, “Os diversity for intrusion tolerance: Myth or reality?,” in IEEE/IFIP International conference on Dependable Systems and Networks, 2011.

[13] A. D. Wood, and J. A. Stankovic, “Denial of service in sensor networks,” *IEEE Computer,* vol. 35, no. 10, 2002.

[14] J. Grand, “Protecting your crown jewels: An introduction to embedded security for hardware-based products,” *Computer Fraud and Security,* vol. October, 2005.

[15] Q. Li, H. Gao, and Z. Jiao, “Hardware threat: The challenge of information security,” in International Symposium on Computer Science and Computational Technology, 2008.

[16] "Common attack pattern enumeration and classification." http://capec.mitre.org/.

[17] M. Rounds, and N. Pendgraft, "Diversity in network attacker motivation: A literature review."

[18] R. Gandhi, A. Sharma, W. Mahoney *et al.*, “Dimensions of cyber-attacks social, political, economic, and cultural,” *IEEE Technology and Society Magazine,* vol. March, 2011.

[19] J. D. Villasenor, "Ensuring hardware cybersecurity," *Issues in Technology Innovation,* 9, Center for Technology Innovation, Brookings, 2011.

[20] J. Villasenor, “The hacker in your hardware,” *Scientific American*, no. August, 2010.

[21] "Government security vulnerabilities: The threat of outsourced chips." www.searchsecurity.com.

[22] A. Pellegrini, V. Bertacco, and T. Austin, “Fault-based attack of rsa authentication,” in Design, Automation and Test in Europe conference, 2010.

[23] R. Wojtczuk, and J. Rutkowska, "Attacking smm memory via intel cpu cache poisoning," Invisible Things Lab, 2009.

[24] V. M. Igure, and R. D. Williams, “Taxonomies of attacks and vulnerabilities in computer systems,” *IEEE Communications Surveys & Tutorials,* vol. 10, no. 1, 2008.

[25] P. Meunier, "Classes of vulnerabilities and attacks," *Wiley handbook of science and technology for homeland security*: Wiley, 2008.

[26] S. Christey, "2010 cwe/sans top 25 most dangerous software errors," MITRE & SANS, 2010.

[27] "Owasp top 10 - 2010, the ten most critical web application security risks," www.owasp.org.

[28] K. Bicakci, and B. Tavli, “Denial-of-service attacks and countermeasures in ieee 802.11 wireless networks.,” *Computer Standards & Interfaces,* vol. 31, 2009.

[29] A. Mishra, and K. M. Nadkarni, "Security in wireless ad hoc networks - a survey," *Ad hoc networks handbook*, 2003.

[30] H. D. Lane, "Security vulnerabilities and wireless lan technology," SANS Institute, 2005.

[31] B. Wu, J. Chen, J. Wu, and M. Cardei, "Chapter 12: A survey on attacks and countermeasures in mobile ad hoc networks," *Wireless/mobile network security*: Springer, 2006.

[32] T. Erickson, "User vulnerabilities on the internet; how to mitigate your risk," Capella University, 2005.

[33] E. Zavaleta, j. Pasari, j. Moore *et al.*, “Ecosystem responses to community disassembly,” *Annals of the New York Academy of Sciences*, 2009.

[34] M. Stamp, “Risks of monoculture,” *Communications of the ACM,* vol. 47, no. 3, pp. 120, 2004.

[35] C. Swanston, M. Janowiak, L. Iverson *et al.*, *Ecosystem vulnerability assessment and synthesis: A report from the climate change response framework project at chequamegon-nicolet national forest*, 2010.